We thank the reviewers for their time and helpful remarks. Below we address some key issues and summarize the improvements we will make in the final version of the paper.

Reviewer 1: We thank the reviewer for their positive assessment of the paper and our work. We agree that providing more detail regarding some specific contributions would improve the paper. We will implement the following changes:

- Make better use of supplementary material space and add detailed descriptions of the "prior inflation" scheme and the protocol there, properly referenced from the main text.
- Provide further explanation for the choice of LMH, RMH, and IC inference engines and the reasons that led us focus on their use in this setting.
- Improve the text in Section 5 (Experiments) to highlight the value of the posterior results obtained in the Sherpa simulator.
- Explain how the "interpretable graphs" provided in the paper are interpreted in practice by a domain expert, by providing concrete examples of correspondence between the latent structure in the graph, the actual code implementing the model within the simulator, and the corresponding interpretation in terms of the underlying physics.

We also appreciate the detailed further suggestions about how to improve the text and figures overall, and we will implement these in the final version. We welcome the pointer to the hierarchical implicit models (Tran et al. 2017).

Reviewer 2: We thank the reviewer for their positive remarks and encouragement. The reviewer raises the issue that the paper's description of the framework is abstract and omits some of details. We will address the following points:

- Provide a concrete introductory example of the PPL in supplementary material. In addition to this, also note that our intention has been to make the actual Python/C++ code of the PPL and the simulator public by providing URLs in the final version of the paper. We believe that this will help the reader see and inspect the actual system in practice.
- IC inference engine is implemented in a way that is automatic, that is, the structure of the neural network is created by the system on-the-fly based on the simulator address space and the user does not have to implement a neural network for proposals. We will explain this in more detail, using the same PPL example we will introduce (previous item). We will also clearly specify the work needed by a user to connect an existing simulator for cross-platform execution.
- The PyTorch PPL serves the purpose of providing the basic PPL constructs (implementing, e.g., *sample* and *observe* statements, trace recording, inference engines) that are called from the existing simulator (implementing the probabilistic model) as a side-effect of random number sampling, through the protocol described in the paper. We will make this clearer in the text and in the figures.

We appreciate the other suggestions for improvement (text and references), which we will implement in the final version.

Reviewer 3: We thank the reviewer for their positive assessment together with bringing up very important points that need to be addressed to provide more technical detail in the paper. We will address the following key issues raised by the reviewer:

- The "discovery of new fundamental physics" has been the main motivation of the particle physics collaborators participating in our work. Ultimately the inference stage of the approach described here would be run on real collision data, which is largely not available for use outside LHC collaborations. Now that our approach has been demonstrated in this paper, we will work with these collaborations to implement it on a full LHC physics analysis reproducing the efficiency of point-estimates, together with the full posterior, so that this can be exploited for discovery. Note that the simulators we use also contain models of processes beyond the current Standard Model of particle physics, and the use of these parametrized simulators for building the model does not limit the ability to discover new physics when applied to real collision data in the inference stage. We will provide these explanations in the paper to support our claim about the potential future value of our work.
- The comparison between the PPL protocol and ONNX was meant as an analogy to help understand our contribution, i.e., we are introducing a project of interoperability between probabilistic programming languages by allowing a language-agnostic exchange of existing models (simulators) and PPLs. We will improve the text to clarify this.
- We will provide a more detailed introduction to our PPL using a concrete example (see Reviewer 2) and API signatures, and also improve the description of what is meant by implementing a universal PPL.
- Thank you for pointing out the reparametrization examples in Stan, we will improve our text to omit the unnecessary generalization you pointed out in line 238.
- Our PPL and protocol support conditioning on arbitrary addresses which are not restricted to be the last in the execution trace. In the Sherpa example the observations are in the end due to the conventions of using this simulator and the physics setting of conditioning on calorimeter data, which come at the end. We will provide a clear explanation of the conditioning mechanism and its implementation details.